Quantum cryptanalysis of ECC[1]

Gustavo Banegas[2]

February 22, 2021

[1] Joint work with Daniel J. Bernstein, Iggy Von Hoof and Tanja Lange to be presented at CHES2021

[2] INRIA & LIX - École polytechnique, France
gustavo@cryptme.in

Outline

Introduction

ECC Operations

Quantum Computation

Quantum Algorithms
- Shor's algorithm

Table of Contents

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical

In other words..

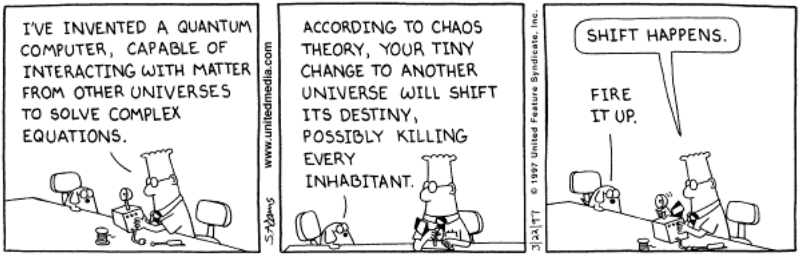


Table of Contents

Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;

Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;
- We use polynomial representation and the operations are in $\mathbb{F}_2$ since $\mathbb{F}_{2^n} \cong \mathbb{F}_2[z]/(m(z))$, where $m(z)$ is an irreducible polynomial of degree n;

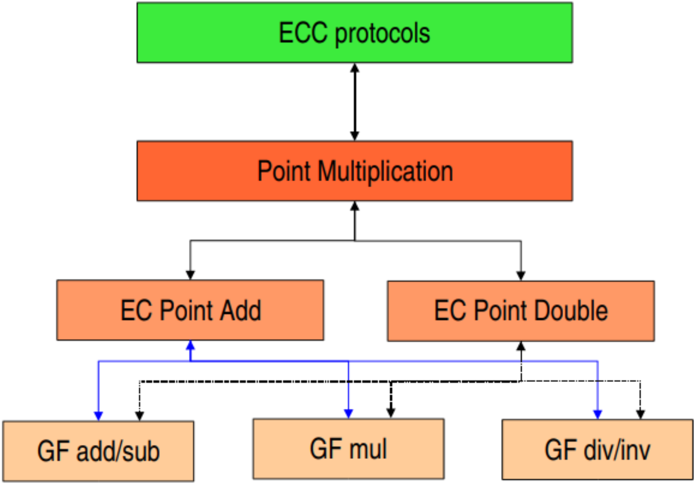
Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;
- We use polynomial representation and the operations are in $\mathbb{F}_2$ since $\mathbb{F}_{2^n} \cong \mathbb{F}_2[z]/(m(z))$, where $m(z)$ is an irreducible polynomial of degree n;
- All computations are done mod $m(z)$.

Introduction to Binary ECC

Basic overview of operations



Introduction to Binary ECC

Hardness of ECC

- Alice and Bob agrees in the same point P over a curve;

Introduction to Binary ECC

Hardness of ECC

- Alice and Bob agrees in the same point P over a curve;
- Alice selects a secret integer α and Bob selects an integer β;

Introduction to Binary ECC

Hardness of ECC

- Alice and Bob agrees in the same point P over a curve;
- Alice selects a secret integer α and Bob selects an integer β;
- Then, they calculate and tell each other $P_\alpha = [\alpha]P$ and $P_\beta = [\beta]P$;

Introduction to Binary ECC

Hardness of ECC

- Alice and Bob agrees in the same point P over a curve;
- Alice selects a secret integer α and Bob selects an integer β;
- Then, they calculate and tell each other $P_\alpha = [\alpha]P$ and $P_\beta = [\beta]P$;
- Finally, they calculate their shared point $P_{\alpha\beta} = [\alpha \cdot \beta]P = [\alpha]P_\beta = [\beta]P_\alpha$.

Table of Contents

How a quantum computer works?

- It perform computations based on **probabilities** of an object's state before it is measured;

Introduction to Quantum Computing

How a quantum computer works?

- It perform computations based on **probabilities** of an object's state before it is measured;
- We can change the probabilities of a **state**;

Introduction to Quantum Computing

How a quantum computer works?

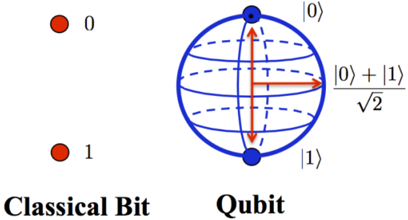
- It perform computations based on **probabilities** of an object's state before it is measured;
- We can change the probabilities of a **state**;

Quantum Computation - qubits

Qubit vs Classical bit

Quantum Computation - qubits

Qubit vs Classical bit



$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

$$\alpha\,|0\rangle + \beta\,|1\rangle\,,$$

$$|\alpha|^2 + |\beta|^2 = 1$$

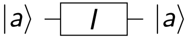
.

Measure quantum state

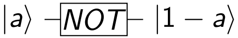
Measuring collapses the state.

Quantum gates

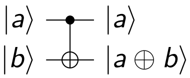
Identity gate:



NOT gate:

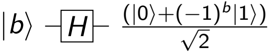


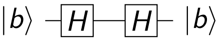
CNOT gate:



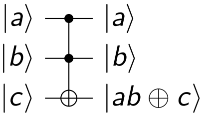
Hadamard Gate:

- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$





Toffoli gate:



n-Qubit system

Definition

$|\psi\rangle \in \mathbb{C}^2$ such that $|| \, |\psi\rangle \, || = 1$,

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

where

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1.$$

Example 2-qubit system

- 4 basis states: $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$,$|1\rangle \otimes |0\rangle$, $|1\rangle \otimes |1\rangle$.
- It is common to use just: $|0\rangle\,|1\rangle$,$|10\rangle$

Quantum computation and reversibility

Reversibility

Quantum evolution is unitary (or any operation that changes the state needs to be unitary);
Unitary means:

$$UU^\dagger = U^\dagger U = I$$

Quantum computation and reversibility

Reversibility

A unitary transformation taking basis states to basis states must be a permutation.

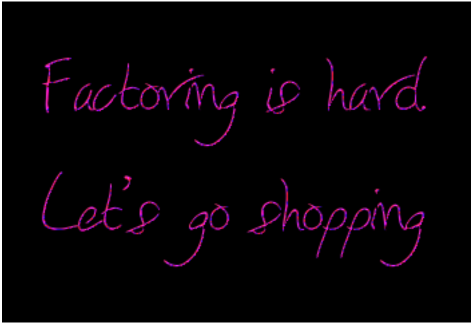
if $U|x\rangle = |u\rangle$ and $U|y\rangle = |u\rangle$, then $|x\rangle = U^{-1}|u\rangle = |y\rangle$.

Therefore quantum mechanics imposes the constraint that classically it must be reversible computation.

Table of Contents

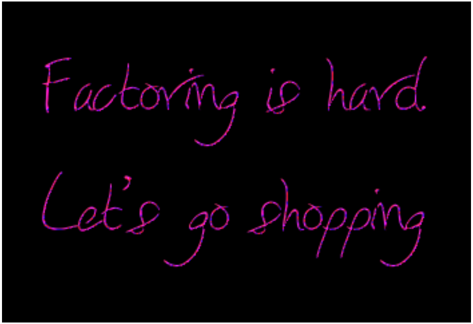
Factoring prime numbers

Factoring Integers with Shor's algorithm



Factoring prime numbers

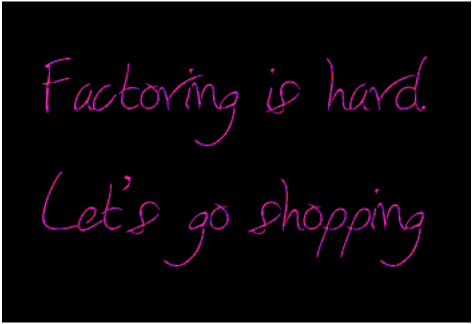
Factoring Integers with Shor's algorithm



- Develop by Peter Shor in 1994;

Factoring prime numbers

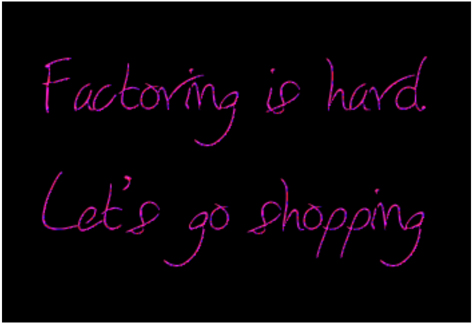
Factoring Integers with Shor's algorithm



- Develop by Peter Shor in 1994;
- Brings apocalypse to cryptography;

Factoring prime numbers

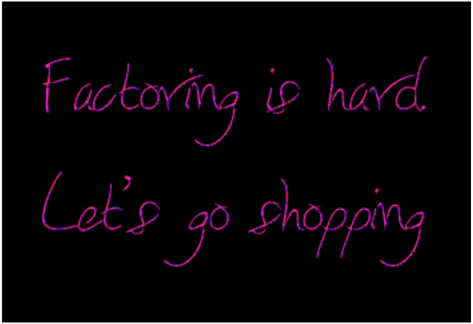
Factoring Integers with Shor's algorithm



- Develop by Peter Shor in 1994;
- Brings apocalypse to cryptography;
- It breaks RSA, ECDSA and DSA;

Factoring prime numbers

Factoring Integers with Shor's algorithm



- Develop by Peter Shor in 1994;
- Brings apocalypse to cryptography;
- It breaks RSA, ECDSA and DSA;
- How many qubits and gates do we need to run Shor's algorithm?

Shor's algorithm

In summary Shor's algorithm has two parts:

- A reduction of the factoring problem to the problem of **order-finding**, which can be done on a classical computer;

Shor's algorithm

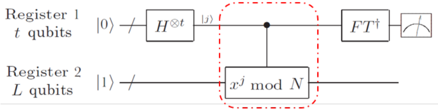
In summary Shor's algorithm has two parts:

- A reduction of the factoring problem to the problem of **order-finding**, which can be done on a classical computer;
- A quantum algorithm to solve the **order-finding** problem.

Shor's algorithm

A toy example can be when we have $N = 15$. Let's see how Shor's algorithm works:

1. Select an arbitrary number, such as $a = 2$ (< 15)
2. $gcd(a, N) = gcd(2, 15) = 1$
3. Find the period of function $f(x) = a^x \mod N$, which satisfies $f(x + r) = f(x)$;
4. Get $r = 4$ through the circuit below;
5. $gcd(a^{\frac{r}{2}} + 1, N) = gcd(5, 15) = 5$;
6. $gcd(a^{\frac{r}{2}} - 1, N) = gcd(3, 15) = 5$;
7. For $N = 15$, the two decomposed prime numbers are 3 and 5.



Ressource Estimation

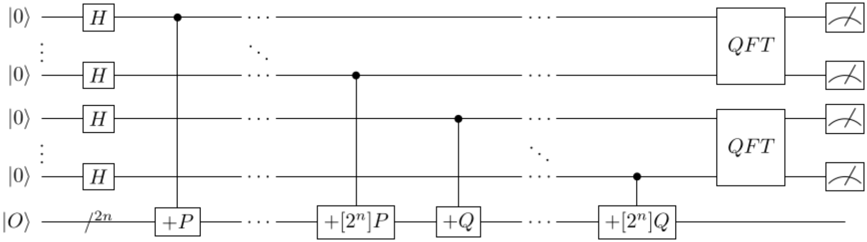
Break RSA (Integer Factoring)

From Gidney & Ekerå(2019)[3] uses "$3n + 0.002nlg(n)$ logical qubits, $0.3n^3 + 0.0005n^3lg(n)$ Toffolis, and $500n^2 + n^2lg(n)$ measurement depth to factor n-bit RSA integers"

RSA Bits	Qubits	Toffoli + T Gates (billions)
1024	3092	0.4
2048	6189	2.7
3072	9287	9.9

[3]Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. arXiv preprint quant-ph/1904.09749, 2019. https://arxiv.org/abs/1905.09749

Shor's circuit for finding elliptic curve discrete logarithm



Shor's circuit for finding elliptic curve discrete logarithm

- Implementation (Quantumly) of Inversion using GCD and FLT (Fermat's little theorem);

[4] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binaryfinite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for finding elliptic curve discrete logarithm

- Implementation (Quantumly) of Inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[4];

[4] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binaryfinite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for finding elliptic curve discrete logarithm

- Implementation (Quantumly) of Inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[4];
  - The GCD-based inversion performed better in number of qubits and gates.

[4] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binaryfinite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for finding elliptic curve discrete logarithm

- Implementation (Quantumly) of Inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[4];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";

[4] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binaryfinite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for finding elliptic curve discrete logarithm

- Implementation (Quantumly) of Inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[4];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";
- Present the a quantum version of "window" addition;
- Q# implementation of Karatsuba and other functions.

[4] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binaryfinite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

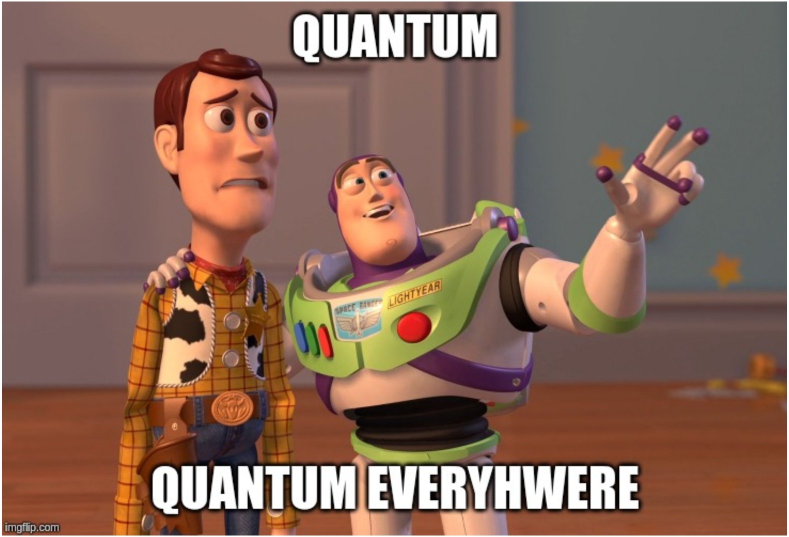
Ressource Estimation

Break Binary ECC (DLP)

From Banegas, Bernstein, von Hoof and Lange(2021)[5] we have that for breaking binary ECC we have $7n + \lfloor \log(n) \rfloor + 9$ qubits, $48n^3 + 8n^{\log(3)+1} + 352n^2 \log(n) + 512n^2 + O(n^{\log(3)})$ Toffoli gates and $O(n^3)$ CNOT gates (More details in the presentation at CHES2021).

		Single step			Total
n	qubits	TOF gates	CNOT gates	depth upper bound	TOF gates
163	1,157	893,585	827,379	1,262,035	293,095,880
233	1,647	1,669,299	1,614,947	2,405,889	781,231,932
283	1,998	2,427,369	2,358,734	3,503,510	1,378,745,592
571	4,015	8,987,401	9,080,190	13,237,682	10,281,586,744

[5] Banegas, G., Bernstein, D. J., van Hoof, I., Lange, T. Concrete quantum cryptanalysis of binary elliptic curves. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1)


QUANTUM
LIGHTYEAR
QUANTUM EVERYHWERE
imgflip.com

Other Quantum algorithms

- Simon's Algorithm (QFT);
- Ambaini's Algorithm (Element disticness);
- Claw finding Algorithm;
- Kuperberg's Algorithm (dihedral hidden subgroup problem);

Questions

Thank you for your attention.
Questions?
gustavo@cryptme.in